IDENTIFICATION OF ANONYMOUS USERS IN TWITTER

İnanç Arın

Submitted to the Graduate School of Engineering and Natural Sciences
in partial fulfillment of the requirements for the degree of
Master of Science

Sabancı University

August, 2012

IDENTIFICATION OF ANONYMOUS USERS IN TWITTER

Approved by:

Assoc. Prof. Dr. Yücel Saygın (Dissertation Supervisor)	.....
Assoc. Prof. Dr. Berrin Yanıkoğlu	.....
Asst. Prof. Dr. Hüsnü Yenigün	.....
Asst. Prof. Dr. Mehmet Ercan Nergiz	.....
Assoc. Prof. Dr. Tonguç Ünlüyurt	.....

Date of Approval: .....

IDENTIFICATION OF ANONYMOUS USERS IN TWITTER

İnanç Arın

Computer Science and Engineering, Master's Thesis, 2012

Thesis Supervisor: Yücel Saygın

Abstract

Users may have multiple profiles when writing comments, blogs, and tweets on the web. While some of these profiles reveal true identity, the others are created under pseudonyms. This is essential especially in the countries with oppressive governments where activists are writing pseudonymous tweets or Facebook messages. In these countries, government officials discovering the fact that a person is among the activists may have serious consequences, the activist being imprisoned, or even his or her life being jeopardized. Pseudonyms may provide a sense of anonymity, however the writing patterns of an author can provide clues that can be used to link the pseudonymous account to the public account. More specifically, one can look at some features within the text whose author is known, and build a model by using these features to predict whether a given (supposedly) anonymous text belongs to that author or not. In this work, we first demonstrate that a person can be identified as being part of a group by using his/her tweets. We used twitter since it is a popular platform, but the problem is not specific to twitter. We show that through tweets, an adversary can build a classifier from public tweets of known users to match them with pseudonymous twitter accounts. We use a simple vector-space model with tf-idf weights to represent documents and a Naive-Bayes classifer with cosine similarity measure. We show that the problem of matching public and pseudonymous accounts exists in twitter through experiments with real data. We also provide a formalism to describe the problem and based on the formalism we provide a solution to protect the privacy of individuals who would like to stay anonymous when writing tweets.

TWITTER'DAKİ ANONİM HESAPLARIN ORTAYA ÇIKARTILMASI

İnanç Arın

Bilgisayar Bilimleri ve Mühendisliği, Yükseklisans Tezi, 2012

Tez Danışmanı: Yücel Saygın

Özet

Kullanıcılar internet ağı üzerinden yorum, blog ve tweet yazarken birden fazla profile sahip olabilirler. Bu kullanıcıların bir kısmı gerçek kimliklerini ortaya çıkarırken, diğer kısmı profillerini anonim olarak takma isimle oluştururlar. Bu durum özellikle eylemcilerin tweet ya da Facebook mesajlarını anonim olarak yazdığı baskıcı hükümetlerin bulunduğu ülkelerde çok gereklidir. Bu ülkelerde, eylemcilerin arasında bulunan kişilerin ciddi sorunlarla karşılaşabileceği, hapis cezasına çarptırılabilecekleri ve hatta hayatlarının riske atılmış olabileceği gerçeği hükümet yetkilileri tarafından ortaya konulmuştur. Her ne kadar takma isimli bir hesaba sahip olmak belli olaranda gizlilik sağlasa da yazarların yazım şablonu, anonim hesapları ve gerçek hesapları ilişkilendirmek için ipuçları verebilir. Daha belirgin olarak, birileri yazarı bilinen bir yazının özelliklerine bakarak bu özelliklerden bir model oluşturabilir ve bu oluşturduğu modeli verilen başka bir anonim yazının bu yazara ait olup olmadığını tahmin etmek için kullanabilir. Bu çalışmada ilk olarak, verilen bir kişinin tweetlerine bakarak bu kişinin belirli bir grubun parçası olarak tanımlanabileceğini kanıtlayacağız. Biz bu çalışma için çok popüler bir platform olan Twitterı kullanmış olsak da bu problemin sadece Twitter ile kısıtlı olmadığını belirtmek isteriz. Gösterdiğimiz diğer bir nokta, bir aleyhtar gerçek isimli kullanıcıların tweetlerini kullanarak, bu gerçek hesapları anonim hesaplarla eşleştirebilmek

için bir sınıflandırıcı yaratabilir. Biz bu çalışmada Naive-Bayes sınıflandırıcı, kosinüs benzerliği ölçüsü ve dökümanları temsil eden kelimelerin tf-idf ağırlıklarından oluşan bir vektor uzayı kullanmaktayız. Ayrıca problemimizi tanımlayan bir formalizm oluşturarak, tweet yazarken anonim kalmaya davam etmek isteyen bireyler için oluşturduğumuz formalizm üzerinden gizliliği korumaya yönelik bazı çözümler de ortaya koymaktayız.

to my beloved family...

Acknowledgements

This thesis would not have been possible without valuable support of many people. Firstly, I wish to express my appreciation to my thesis supervisor, Assoc. Prof. Dr. Yücel Saygın for his endless assistance for years. He always has been abundantly helpful as an instructor and as a valuable advisor with his patience and knowledge. Besides, Asst. Prof. Dr. Mehmet Ercan Nergiz made an important contribution for my work as he was my other supervisor. I am also thankful to members of my thesis defense committee: Assoc. Prof. Dr. Berrin Yanıkoğlu, Asst. Prof. Dr. Hüsnü Yenigün and Assoc. Prof. Dr. Tonguç Ünlüyurt for their presence and feedbacks. Additionally, I am highly indebted to all my old instructors who contributed to me during my education years.

I have indefinable feelings towards my roommates who are Uğur Bağcı, Yaşar Tüzel, Burak Sezin Ovant, Umut Öztok, Ersin Tanrıverdi and Ahmet Koçyiğit as they will always stay as my brothers. I would like to thank Süha Orhun Mutluergil and Erdi Aker for their precious friendships and supports. I also wish to state my kindest gratitude to Nilay Acat who has a special place in my heart.

I would like to extend my sincere thanks to Sakıp Sabancı who has created all these opportunities for us.

Last but not least, I want to express my special appreciation and thanks to my beloved family as they have always supported and encouraged me. I am always proud of being a part of this family.

Contents

List of Figures

List of Tables

1 Introduction

Privacy preserving data management has been an active research area for the past few decades. One of the main problems of privacy preserving data management is to prevent linkage attacks on data to be published. Although privacy preserving data publishing has been studied a lot, there are still privacy leaks due to release of different and more complex data sets. For example in August 4, 2006, American Online (AOL) published a data set containing the search logs of about 650.000 users over a 3 months period [2]. Actually, this data set has been published so that researchers can experiment with data mining techniques for search query optimization and other similar purposes. AOL de-identified the data by deleting the direct identifiers and the IP addresses, however, there was still a scandalous privacy leak that made it to the New York Times damaging the reputation of AOL. The privacy leak was demonstrated by a curious journalist to identify a user just by combining the search keywords of that user with some background knowledge. An AOL user with pseudonym 4417749 (Thelma Arnold) was identified by her searches. The search keywords of Thelma Arnold gave hints on the place she lives and her relatives since she was searching people with surname Arnold [5] together with her interests. Although AOL had removed the data by accepting that sharing this data was a mistake [17], the data can still be downloaded from some internet mirrors. Many search engines such as Google, and Bing keep logs of our queries in their databases. Due to privacy concerns, Google has announced that they anonymize their query logs after a period of time, but anonymization of web logs is still an open research question. For example authors in [18] show that combinations of small pieces of search logs can be efficiently used for identification of individuals like in the AOL case.

In general, a user's activities on the web can actually be used to distinguish that user from others. Social interactions, visited web sites, communications etc can be defined as user activity. Generally all of these activities come into consideration where each of them may construct a path to make a model of individuals. For example in [29] authors show that syntactic information can be used to identify the author of a given text. In experiments, it is proven that length of sentences and paragraphs, number of words in a sentence or paragraph, paragraph style like indentation style and the spacing between paragraphs can be used as key points to distinguish each user. Available text written by web users has been increasing especially with the usage of social networks. Consequently social networks become an important area for researchers to work. By using social media, users tend to correspond with each other and share their comments and ideas. Newsgroups, and blogs have been popular platforms where users write about themselves or comment about popular issues. With social media this phenomena has become more widespread and even changed the way the information is collected and disseminated. People use social media not only for their daily social life but also for some critical social or political events. In recent years, people have been started to get together on social platforms to show their reactions, to become organized and to exchange and share information between each other. We have seen that social media has played an important role in the Arab Spring where Facebook and Twitter have been used by the activist to inform the world about what has been happening and to organize protest meetings. In social media, people may have multiple profiles due to various reasons. While some of these profiles public, revealing the actual identity, the others are created under pseudonyms to write freely and anonymously. In the countries with oppressive governments it is vital for the activists to stay anonymous when writing tweets or Facebook messages.

Pseudonyms in social media provide a sense of anonymity. However, writing patterns of people may provide clues that can be used to match a pseudonym with a public account. A sophisticated attacker may use data mining techniques to build a model to predict the author of (supposedly) anonymous text. We claim that it is possible to identify a pseudonym with large certainty by using repeating misspelling and the frequently used words. In this work, in order to demonstrate our claims, we focus on twitter and show that through tweets, one can

extract some features that can distinguish a user from the others. These features could be the writing mistakes, or specific words used that are topic independent. We treat the collection of the tweets by the same user as a document and represent these documents with vector-space model with tf-idf weights and use cosine similarity to identify users.

Doubtless, people make some errors while they performing such actions on the online environment. As it is stated in [16], during their activities, users may fill wrong forms or they may type wrong URL. People usually experience various number of problems during their internet activity and they are forced to make some errors with a good or bad grace. Here, we also focus on a specific type of error which is their spelling errors. Millions of people share their ideas and emotions via social network like Twitter, Facebook everyday. While they are sharing such emotions and ideas, they type lines of text consistently with their spell errors. We can make an model of each user by detecting specific spell error types with their statistical information and we claim that it is possible to identify owner of a text by using this model. When we have a model of a public account, then a pseudonym tweet (or some other text) can be matched with public accounts through a classifier.

Since it is not possible to find a data set with public and pseudonymous accounts, we simulate this scenario by separating the tweets of a user into two disjoint sets, and do this for all the users. One set of tweets becomes public account of a user and the other one becomes its pseudonym. Then, we try to re-match a given pseudonym with its true public account by considering the features we mentioned above. Our experiments showed that around 90% of the case, we can re-join the disjoint sets with the matching account. This shows that the problem of matching public and pseudonymous accounts exists in twitter which is the main contribution of this work.

2 Preliminaries and Background

In this section, we formally introduce some preliminary notation which will be used to solve the problem. Firstly, we will introduce *misspelling operations* as we will use them for detecting error types in a misspelled word. *Minimum Edit Distance* algorithm will be provided for finding distance between strings. Then, *text categorization* and *classification* techniques will be mentioned to explain constructing learning models and application of them. *Document indexing* process is needed for converting text documents into representable formats and *Cosine similarity* will be used to calculate similarity between documents.

2.1 Misspelling Operations

Misspelling operations are defined in [41] as processes that can be applied to any kind of text and can provide a path between word and misspelled word. When we consider the nature of misspellings, we can divide them into three different categories:

1. Insertion: misspelled string has one more character that correct string does not have.
2. Omission: misspelled string has an absent character that correct string has.
3. Substitution: one character is replaced by a different one in the correct string.

It can be useful to give examples to these error types. When a user types *ther* instead of *the*, it is an insertion error. If a user types *th* instead of *the*, then we evaluate this error type as deletion. Lastly, if a user types *thw* instead of *the*, then this error is cagetorized as substitution.

Misspelling operations can be used to identify individuals with collecting all misspell errors of each user and constructing a learning model through this information. Misspell

errors play an important role in our word, therefore this topic will be discussed in later section in more detail.

2.2 Minimum Edit Distance Techniques

As we said before, misspelling operations are crucial for our claim; however apart from this, we also need to find minimum edit distance between a misspelled word and a correct version of this word. In this point, we need to mention about minimum edit distance concept as Kukich defines Minimum Edit Distance in [20] as the minimum number of misspelling operation 2.1 (insertion, omission, and substitution) to convert from one string to another. Firstly, the *minimum edit distance* algorithms were implemented by Damerau and Levenshtein; however, in this paper the algorithm which is implemented by Levenshtein, as the most common distance algorithm, is used as Minimum Edit Distance Technique. Baake defines Levenshtein distance in [3] as a special technique which plays an important role for comparison of symbolic sequences in many well known areas like linguistic, genome research etc. Pseudocode for Levenshtein Distance, which is taken from [7], can be found in the algorithm 1.

The space and time complexity of the algorithm is $O(nm)$. However space complexity can be improved since we only need for previous row and columns instead of keeping all of them.

Here, there is an example for finding Levenshtein Distance (LD) between two strings. In the table 1, we apply the Levenshtein algorithm to find the edit distance between *sitting* and *kitten*. Firstly, we put default values for $d[i, 0] = i$ where i is $1 \rightarrow m$ and for $d[0, j] = j$ where j is $1 \rightarrow n$. Then, in the main loop, for each index of the matrix, we check whether characters of two strings are same. As it is stated in the algorithm, if these characters are same, then we determine the current index $d[i, j]$ exactly same as $d[i-1, j-1]$. Otherwise, we attribute current value to $minimum(d[i-1, j]+1, d[i, j-1]+1, d[i-1, j-1]+1)$. When we look at the different character indexes in the table, we determine $d[1, 1] = 1$ since the minimum value is $d[0, 0]$ which means it is a substitution between s and k. Then, we find a difference in $d[5, 5]$ such that the minimum value holds in index $d[4, 4]$ which again means a substitution between e and i. Lastly, the final difference occurs in $d[7, 6]$ such that the minimum value

Algorithm 1 Levenshtein Distance Algorithm (*char s[1..m], char t[1..n]*)

```
1: declare int d[0..m, 0..n]
2: set each element of d to zero
3: // for all i and j, d[i, j] will hold the Levenshtein distance between
4: // the first i characters of s and the first j characters of t
5: for i = 1 → m do
6:     d[i, 0] ← i
7: end for
8: for j = 1 → n do
9:     d[0, j] ← j
10: end for
11: for j = 1 → n do
12:     for i = 1 → m do
13:         if s[i] = t[j] then
14:             d[i, j] ← d[i − 1, j − 1]
15:         else
16:             d[i, j] ← minimum(d[i − 1, j] + 1, d[i, j − 1] + 1, d[i − 1, j − 1] + 1)
17:             // d[i − 1, j] + 1 means a omission
18:             // d[i, j − 1] + 1 means an insertion
19:             // d[i − 1, j − 1] + 1 means a substitution
20:         end if
21:     end for
22: end for
23: return d[m, n]
```

is $d[6, 6]$ meaning that there is an insertion of g in the current point. Consequently, as it is explained, the $LD(sitting, kitten) = 3$ since there should be one substitution(s,k), one other substitution(i,e) and one insertion(g) which means 3 transactions are needed to convert *kitten* to *sitting*.

In this work, we use Minimum Edit Distance Technique (Levenshtein Distance) since it provides some additional information about misspelled words of a user. As it is said before, we determine what kind of misspelling error(s) are done by user with their detailed information and we also calculate the edit distance between the misspelled word and the correct word. This distance information can also be used for a modeling operation of a specific user. In this point, we need to consider following question: When a user makes a mistake when typing, how can we find out the correct word that user implies. In other words, we need to understand the real word to obtain edit distance and misspelled error types (insertion, omis-

		k	**i**	**t**	**t**	**e**	**n**
	0	1	2	3	4	5	6
s	1	1	2	3	4	5	6
i	2	2	1	2	3	4	5
t	3	3	2	1	2	3	4
t	4	4	3	2	1	2	3
i	5	5	4	3	2	2	3
n	6	6	5	4	3	4	2
g	7	7	6	5	4	4	3

Table 1: An example to find edit distance between two words

sion, substitution). For this purpose, we firstly check whether a word is typed correctly by searching this word in the spell checker of Google. If it exists in the dictionary, it means that this word was written correctly; otherwise it means, willingly or unwillingly, user made a mistake when typing the specified word. If we understand that the word is a misspelled word, then Google provides some suggestions for the implied (correct) word. At this juncture, we assume that the first suggestion, which is provided by Google, is the real implied word by user. Then, we take this word and start to find our spell error types. More detailed information will be explained in section 4.1.

2.3 Text Categorization

Text Categorization (or *Text Classification*) can be evaluated as the activity of labeling of natural language texts as some predefined categories. More formally, it can be defined as the process of assigning a boolean value to each pair of $(d_j, c_i) \in D \times C$ where D is set of documents and C is set of predefined categories $C = \{c_1, c_2, \ldots, c_n\}$. If (d_j, c_i) is *true*, it means that document d_j can be categorized under category c_i. Otherwise, the situation that *false* is assigned to the a pair (d_j, c_i) indicates that the document d_j does not belong to the category c_i. All these information is indicated in [34] as the following crucial points are also referenced in the same paper. In the concept of Machine Learning area, classification problem can be defined as a supervised learning problem because the execution part is mainly composed of learning process by using the knowledge of category information of training set for the prediction of test set. In this point, definitions of training and test sets are given below as they

were defined in [34].

- *a training (or validation) set* $TV = \{d_1, d_2, \ldots, d_{|TV|}\}$. This is the set of documents observing the characteristics of which the classifiers for the various categories are inductively built;

- *a test set* $Te = \{d_{|T|+1}, \ldots, d_{|\Omega|}\}$ where $\Omega = \{d_1, \ldots, d_{|\Omega|}\}$ is *initial corpus* that documents previously classified under the same set of categories. *Test set* will be used for the purpose of testing the effectiveness of the classifiers.

2.4 Document Indexing

One other important point, which emphasized by authors in [34], is *document indexing* process. Document indexing is crucial for machine learning or data mining tasks since a classifier or a classifier building algorithm cannot directly process a text. For this reason, we need an indexing technique such that it maps a text d_j into a representable format for classifiers to make this text d_j eligible for being interpreted by classifiers and algorithms. Needless to say, documents in both training and testing set should be converted into representable format by an indexing procedure for the certain reason. In this concept, a text document d_j *is represented as a vector of weights* $d_j = < w_{1j}, w_{2j}, \ldots, w_{|T|j} >$ where T is set of terms or features and $0 \leq w_{kj} \leq 1$ which indicates how much term t_k plays an important role semantically in document d_j.

One other technique for indexing procedure is keeping *phrases* rather than individual words as indexing terms in [15, 33, 37]. However, the experimental results show that the idea of phrases is not so encouraging.

As it is pointed before, for a term t_k, the range of its weight is $0 \leq w_{kj} \leq 1$ (however, there is an exception in [23]) and in some studies like [1, 19, 22, 24, 28, 27, 32, 35, 33], authors preferred to assign a binary value to the w_{kj} in the indexing system. For this case, w_{kj}=1 means the presence of the term t_k in the document d_j and w_{kj}=0 means the absence of the

term t_k in the document d_j.

In the case of non-binary indexing technique, which is more widely used in *Information Retrieval* applications, generally *tfidf* weighting function is used to create vector of weights for each document as it is stated in [30]. The *tfidf* weight (term frequencyinverse document frequency) is a statistic information which reflects how important a word is to a document in a set of collection of documents. The formula of tfidf is given below:

$$tfidf(t_k, d_j) = \#(t_k, d_j) \cdot log \frac{|Tr|}{\#T_r(t_k)}$$

where $\#(t_k, d_j)$ is the number of occurrences of t_k in d_j, $|Tr|$ is the size of the document corpus, $\#T_r(t_k)$ is the number of documents in Tr that t_k appears in. When we analyze characteristics of the formula, if a term t_k occurs frequently in some specific document d_j, this increases the importance of t_k in d_j, however if t_k also occurs in many documents in the corpus, then importance of t_k for the document d_j decreases according to the *tfidf* formula.

Additionally, these tfidf weights should be normalized by using *cosine normalization* technique to make them fall in [0, 1] interval as normalization way is given in [34]:

$$w_{kj} = \frac{tfidf(t_k, d_j)}{\sqrt{\sum_{s=1}^{|T|}(tfidf(t_s, d_j))^2}}$$

2.5 Dimensionality Reduction

One other important point that we need to discuss about Text Categorization is that *dimensionality reduction*. When we consider the term space for the classification algorithms, high number of term spaces may be problematic like in the LLSF algorithm in [39]. For that reason, usually a reduction in the term vector is needed for the process that reducing $|T|$ to $|T'|$ where $|T'| < |T|$ and T' is named as *reduced term set* as in [34]. For the Term Space Reduction (TSR) problem, one of the well know algorithm, *document frequency* algorithm, is again stated in [34]. The *document frequency* algorithm $\#T_r(t_k)$ keeps only the terms which occurs

highest number of documents with considering the idea that the terms occur frequently in the document corpus are the most valuable terms. It seems like there is a contradiction in this point such that we previously said the terms occur in less documents (in other words the terms which have low to medium frequency [31]) are more informative. However, these two thesis do not contradict each other since there are enormous number of very low frequency words in a large term corpus and when we reduce vector space, only low frequency words are removed from corpus. The informative words which are low to medium frequency still exist in the term corpus. In [40], authors experimentally show that they could manage the reduce vector space by factor of 100 without any loss in effectiveness. Last but not least, before starting the dimensionality reduction process, stop words should be removed from documents to only deal with neutral words as Mladenic has pointed in [26].

2.6 Naive Bayes Classification Model

Naive Bayes algorithm is mainly based on conditional probability. The brief logic behind this model is well known Bayesian formula which takes into consideration frequency and combinations of values in historical data. Naive Bayes aims to calculate the means and variances of the parameters by using training data in a supervised learning way to make a classification model. For this process, Maximum Likelihood method is used as one of the well known methods for parameter estimation.

In our work, one of the preferred classification algorithm is Naive Bayes because of its performance and optimality. The reasons behind its good accuracy are explained in [42], as they show the distribution of dependencies by indicating the distribution of local dependencies of each class and showing how these dependencies work together in some manner.

As Naive Bayes is a probabilistic classifier, it tries to calculate $P(c_i|\vec{d_j})$ which means that a given document donated by a vector of weigths $\vec{d_j} = \left\{w_{1j}, \ldots, w_{|T|j}\right\}$, what is the probability that this document $\vec{d_j}$ belongs to the category c_i, as discussed in [21, 34]. The related formula for Naive is given below:

$$P(c_i|\vec{d_j}) = \frac{P(c_i)P(\vec{d_j}|c_i)}{P(\vec{d_j})}$$

where $P(\vec{d_j})$ is the probability that a randomly chosen document has weights of vector representation as $\vec{d_j}$, $P(c_i)$ means probability that a randomly chosen document belongs to category c_i. To explain $P(\vec{d_j}|c_i)$, we need to make an assumption such that the words in the document are neither dependent on their position nor the length of the document.

$$P(\vec{d_j}|c_i) = \prod_{k=1}^{|T|} P(w_{kj}|c_i)$$

Classification methods contain some main steps for the process. These steps are defined in [6] as the figure 1.

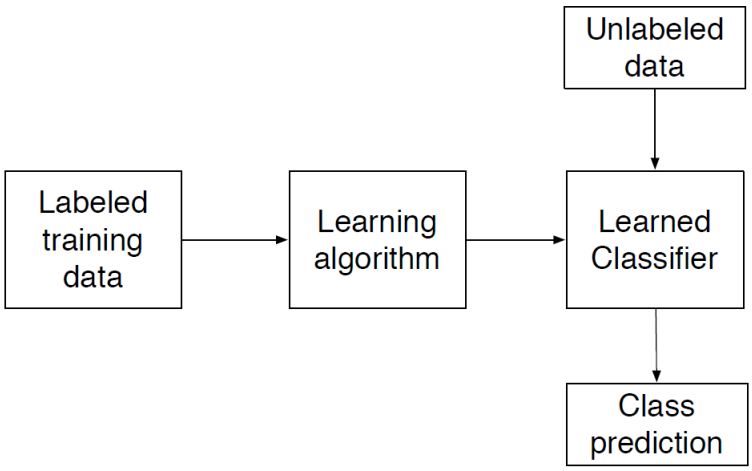

Figure 1: Steps of a classification process

By definition, we need a *labeled training data* for learning algorithm. Then, this algorithm calculates the mean and variance values for each attribute in the dataset; this information becomes as a *learned classifier* (it can be also called as *learned model*). Lastly, when a *unlearned data* comes into question for classification, process ends with a *class prediction* according to the *learned classifier* which we prepared before.

2.7 Cross-Validation

Cross-validation is a well-known technique for the performance evaluation of a classifier. In this technique, data is split into equal sized partitions, and each partition is used same number of times for training process and once for the testing process. As it is explained in detail in [36], assume we split our data into two equal-sized partitions. Firstly, we select one of the partitions as our training data and use other partition as testing data. Then, we change the roles and the second partition becomes a training data while the other one becomes testing data. This technique is named as *two-fold cross-validation* in the literature. To generalize this approach as in [36], we define *k-fold cross-validation* technique as segmenting the data into k equal-sized partitions. While all process is being executed, one partition is selected for testing and the remaining partitions run as training data. This process is applied k times since each of the partitions should become a test data exactly for once. Consequently, all iterations are considered for the final result to evaluate the performance of the classifier.

2.8 Cosine Similarity

We use *Cosine similarity* technique to calculate the similarity function that we describe above. *Cosine similarity* is a similarity technique such that it measures the cosine angle between two documents d_1 and d_2 which are represented by vectors. The maximum value of cosine is 1 (cosine of 0) and the minimum value is -1 that if two vectors are exactly same then their similarity measure is 1.

The main formula for cosine similarity is derived from well known Euclidean dot product formula as given in [36]:

$$A \bullet B = ||A||\ ||B||\ cos(\theta)$$

where $\bullet$ donates vector dot product and $||d||$ denotes the length of vector d. Thus, we can rewrite the formula to obtain cosine value between two vectors:

$$cos(d_1, d_2) = cos(\theta) = \frac{A \bullet B}{||A||\,||B||} = \frac{\sum_{i=1}^{n} A_i \times B_i}{\sqrt{\sum_{i=1}^{n}(A_i)^2} \times \sqrt{\sum_{i=1}^{n}(B_i)^2}}$$

Note that although the range of the cosine of an angle is [-1, 1], the practical range of cosine similarity is [0, 1] in text matching case. The reason behind that both vectors A and B represents $tfidf$ weights which cannot be less than 0.

2.9 K-anonymity

In this part, we will define k-anonymity algorithm for privacy preserving issues. However, before we give the definition of k-anonymity, we need to mention about some concepts which compose the basis of k-anonymity algorithm.

Definition 2.9.1 *Quasi-identifiers (generally abbreviated as QI) are the set of attributes which are not sensitive, however may cause identification of users by using some other resources as defined in [8]. Address, age, sex attributes can be given as examples of quasi-identifiers.*

Definition 2.9.2 *An equivalence class contains set of rows from data where these rows are not distinguishable between each other through a set of attributes (referred in [8]).*

In the light of this information, as defined in [8, 9], we say that a table T is k-anonymous if and only if for each equivalence class E regarding a set of QI, $|E| \geq k$. For instance, when we look at Table [**?**], we see that this table is 2-anonymous since the rows in first equivalence class (first and second rows) are indistinguishable with respect to *Age*, *City* and *Sex* attributes. The other rows in second equivalence class (third and fourth rows) are also indistinguishable between each other with respect to same attributes again.

2.10 RapidMiner

RapidMiner is an open-source data mining tool of which producers represent this tool in[1] with the following two main features: it can be used as an application for data analysis and

[1] http://rapid-i.com/content/view/181/196/

Age	**City**	**Sex**	**Disease**
[20-30]	Izmir	M	Cold
[20-30]	Izmir	M	Leukaemia
[30-40]	Istanbul	F	Broken Leg
[30-40]	Istanbul	F	Flu

Table 2: A table with 2-anonymity

as a data mining engine for the processing its own products. RapidMiner has a world-leading role in its area with a common usage among the people, who is working on data analysis and data mining, under favor of its *data integration*, *analytical tools*, *data analysis* and *reporting* opportunities and also its practical, user friendly *graphical user interface* and numerous other features.

3 Problem Definition

In this section, we are going to give a formal definition of the problem by introducing the methods that we use to prove our claim. Given set of public accounts and another set of pseudonym accounts, we try to determine whether it is possible to find public accounts corresponding to a given anonymous account with a large certainty.

3.1 Formalization of Twitter Accounts

We are going to look at a social media platform such as Twitter where we have a set of accounts C that belong to users, where a user may have multiple accounts. Some of the accounts in C have an account name that could be easily associated with a real person, or the it is publicized that the corresponding account belongs to a real person. We denote these accounts by P. Some of the accounts however are created by users who are activists to express their opinions anonymously, and we denote them by A (for Anonymous Activists). The fact that an individual is among the anonymous activists is private information considering that this may have serious consequences in some countries, or the activist group may be related to a sensitive issue such as gay marriage. We are using the example of activists in twitter, however, the same idea applies to discussion groups on sensitive issues.

We assume that $P \cup A = C$ and $P \cap A = \emptyset$. Each account $a_i \in C$ has an associated set of tweets $T_{a_i} = \{t_{i_1}, t_{i_2}, ..., t_{i_k}\}$ where for each of t_{i_j} for $1 \leq j \leq k$, $t_{i_j} = (w_{i_{j_1}}, w_{i_{j_2}}, ..., w_{i_{j_m}})$, $w_{i_{j_l}} \in W$ for $1 \leq l \leq m$ (m is the number of words in W), W being the universe of words that are used by the Tweeters. Let f be a function that maps an anonymous account to its public account, i.e., $f : A \rightarrow P \cup \{d\}$ where d is the dummy account. If an anonymous account $a_i \in A$ does

not have a corresponding public account, then $f(a_i) = d$. Our claim is that the adversary (In our case the oppressive government trying to uncover the activists) may estimate a function $f' : A \rightarrow P \cup \{d\}$ which is very close to f, such that the probability that the two functions will return the same result is greater than a threshold, i.e., $P(f(a) = f'(a)) > \tau$. Adversary has the tweets of the anonymous and public accounts as a basis for estimating f' for matching a public account with its anonymous account. In fact, we show through experiments that the tweets can actually be used through a similarity function for matching.

In real life, the set of accounts will be evolving, with new accounts being created and tweets being written continuously under the existing accounts. For simplicity, lets assume that the system is frozen and the accounts in A and P are fixed together with the set of tweets that have already been accumulated under these accounts. So there will be no insertion or deletion of the accounts and there will be no more tweets.

Now lets assume that the adversary tries to do the matching on that snapshot of the system. Let s be a similarity function which returns a similarity score for two accounts in terms of the set of tweets written under those accounts. What the adversary can do is to assume that two accounts are candidates for matching if their similarity score is greater than a threshold. There may be a case where many accounts are above the threshold, in this case given an anonymous account a_i the adversary can rank the public accounts with respect to their similarity to a_i from highest to lowest, and consider the top-k most similar ones as the candidates for matching, and those above the similarity threshold within the top-k are the candidates. So, for a given similarity threshold σ, a positive integer k, and an anonymous account a_i, among top-k public accounts a_j where $s(a_i, a_j) > \sigma$ are the candidates for matching for an adversary. Formally, let $g(a_i, \sigma, k) : A \rightarrow P_k(P)$, where $P_k(P)$ is set of all possible subsets of P with cardinality k, is a function such that for a given account $a_i \in A$, a similarity threshold σ, and a positive integer k, it returns a set of accounts $B \subseteq P, |B| \leq k$, where $B = \{b_i \in P | s(a_i, b_i) > \sigma)\}$. We claim that through a good similarity function $P(f(a) \in g(a, \sigma, k)) > \delta$ for even very high similarity thresholds and show this through experimental results on real tweet data.

3.2 Identification Methods

We looked at two different identification techniques; Errors made by users when typing and TF-IDF based cosine similarity measures.

3.2.1 Identification Through Misspellings

We claim that misspellings are a strong identifier for the users and therefore can be used to match an anonymous account with its public account. As we mentioned in Section 2.1, misspelling operations are represented in [41] as processes that can be applied to any kind of text and can provide a path between word and misspelled word. We can divide them into three main categories:

1. Insertion: the misspelled string has a character that the correct string does not have. For instance, user types *ther* instead of *the*.

2. Omission: the misspelled string has an absent character that the correct string has. For instance, user types *th* instead of *the*.

3. Substitution: one character is replaced by a different one in the correct string. For instance, user types *thw* instead of *the*.

We use all these three categories to model the misspellings in the tweets of a specific user. As learning features we use the misspelled word, the correct word, and also to learn which of these error types occurred together with their frequency. Additionally, we keep track of not only misspelling error types but also specific character that leads to occurrence of this error. In other words, we will construct a learning model -by using Naive Bayes modeling technique - for each user to apply this model in the classification algorithm. Assume we have d misspellings in our corpus, and assume there are d' corresponding correct words. Then feature vector contains $(d + d' + 4 * c)$ elements where c is the number of different characters used by all users. $4 * c$ comes from we have for different types of features which are insertion, deletion, substitution1 and substitution2 as in Table 4. For that reason feature vector becomes as follows:

$$featureVector = \begin{pmatrix} misspelledWord(w_1) \\ misspelledWord(w_2) \\ . \\ . \\ correctWord(w_1) \\ correctWord(w_2) \\ . \\ . \\ insertion(c_1) \\ insertion(c_2) \\ . \\ . \\ deletion(c_1) \\ deletion(c_2) \\ . \\ . \\ substitution1(c_1) \\ substitution1(c_2) \\ . \\ . \\ substitution2(c_1) \\ substitution2(c_2) \\ . \\ . \end{pmatrix}$$

3.2.2 Identification Through TF-IDF Based Weighted Cosine Similarity

Identification through typos and misspellings is based on a single word and which kinds of errors does the user do for a given word. We also considered the set of tweets of users as a bag of words and looked at how much this set can identify the user. In order to do that, we used the TF-IDF weighting scheme where words which are used a lot by the user but not by other users are given a higher weight. What an adversary can do is to construct the TF-IDF vectors (with size of $|W|$ where W is set of all words and each index represents the tf-idf weight of term w_i) of the users in P and A in terms of their set of tweets and apply the cosine similarity of the accounts using the TF-IDF vectors.

$$featureVector = \begin{pmatrix} tf-idf(w_1) \\ tf-idf(w_2) \\ . \\ . \\ tf-idf(w_{|W|}) \end{pmatrix}$$

3.3 Possible Solutions for Identification

In this section, we suggest some solutions according to the observations that we make during our work. In the Section 4, we try to prove our claim with some number of experiments while we also explain the reasons behind our claim. We recommend the solutions based on the reasons which we also explain the cause and effect relation between these reasons and our claim in the following parts of the thesis.

In Section3.2, we mentioned that we proposed two methods for identifications of users as well as we will propose two possible solutions for these identification methods, one solution for each method, to help keeping privacy of users on social web. At this point, we are in need of clarifying a subject which underlies this thesis. Although we propose two methods for identification, our main goal is not helping the adversaries who identify users for different purposes. On the contrary, our aim is to experimentally show that users can be identified by using some data mining techniques; whereas users may think that their privacy is in safe just by changing user name or IP but the truth is different. In this sense, we suggest solutions against two methods we proposed, so that anonymous users cannot be matched with their public accounts with high accuracy.

Firstly, as we pointed in Section3.2.1 we can detect anonymous accounts by modeling their spelling errors. Classifier learns each user's misspell behavior with their characters, error types, etc. As a result of this process, classifier may notice some kind of misspelled word or error type is a typical behavior of a specific user a_i. The suggested solution is that we may design a system such that this system may control tweets just before posting operation of user. If the system finds a spell error which is special to the this user (or which can cause identification of anonymous user) then it may automatically correct this misspelled word.

Our second solution is against the identification through tf-idf weight based cosine similarity method. Logic behind this method is the identification of users through the words with high tf-idf value for this user. High tf-idf value of a word w for a user u means that this word w is used frequently by user u and other users do not use w as much as u does (see Section 2.4).

In other words, cosine similarity technique finds similarity scores between users and it makes use of vector representations of documents where these vectors are composed of tf-idf values of words. Therefore, we may again develop a system that detects the suspicious words of which tf-idf values are high for a user and can cause detection of user for this reason. The system may warn the user to change the aforementioned word or it may automatically change this word with its synonyms for which tf-idf value does not endanger the user.

Lastly, we need to point that we are still working on these solutions and they may be improved after some number of experiments and observations. Performance and effectiveness of these solutions are considered as a future work as it will be mentioned later.

4 Experimental Evaluation

In this work, we firstly make some number of experiments to show that tweets with anonymous authors could be linked to other tweet accounts with known user identities via specific features that appear in the written text. We use text classification as a tool for linking different user identities and consider the spelling mistakes and some other features as attributes for classification for the evaluation of the first method which is explained in Section 3.2.1. On the other hand, for the second technique, we used cosine similarity technique as we share results of the experiments below.

We experimented with a Twitter data that was made available by the CAW 2.0 (Content Analysis for WEB 2.0) workshop organized at WWW 2009 conference. The data could be downloaded from $http://caw2.barcelonamedia.org/node/24$. This data set contains about 900K posts of about 27K users. The average number of tweets per account is about 36, and the maximum number tweets observed for a user is 223. Additionally, 5303 users have more than 50 tweets posted in the data. There are also users with very few tweets, therefore we do not consider those users in our experiments. The tweets were collected between the period of October 2006 and January 2009. For all tweets posted by a user, it is possible to find a timestamp information that indicates the exact posting time of the relevant tweet.

4.1 Experiments on the Typos and Misspellings

In this section, we define the steps of the algorithm, mentioned in Section 3.2.1, in a development sequence. We firstly need to train that data to make model of each user. For this purpose, for each word in each tweet, we check whether this word is typed correctly. In this

process, as it is stated in previous sections, we use Google spell checker to understand correctness of a word. Actually, there is a Java API which calls Google's spell checker service from Java applications. If the word which we check is typed incorrectly, then spell checker suggests some number of words with their ranking value and we pick the first suggested word as default. In other words, we assume that if some written word has a spelling error then the highest ranked word by spell checker is the real intended word by user. This process is executed for each tweet, and it is possible to analyze which words are written incorrectly, which misspelled words are used instead of corresponding correct words, the edit distance between correct and misspelled word and the frequency of the occurrence of an error. Moreover, we can also obtain specific error types with specific letter or characters for each misspelled word. In this point, we are able to construct a Naive Bayes model for spelling error behavior of a specific user by using these analytical information we obtain.

After we construct our model, when an unknown tweet comes into question we again firstly detect misspelled words in this tweet, the edit distance between correct and misspelled word, and the error types which occurred in these words, then we are ready to identify the author of this tweet by using a categorization technique with the model we constructed before. For categorization and model construction processes, different kinds of categorization and modeling techniques can be applied. However in our experiments we applied Naive Bayes classification technique since it is a probabilistic classifier and it returns a probability score for each category.

For the categorization process explained above, we use a data mining tool RapidMiner which is mentioned in Section 2.10. A caption from the categorization process of RapidMiner is given in Figure 2. When we look at the overall of the process, there are 5 operators which are *Read Excel*, *Read Excel(2)*, *Naive Bayes*, *Apply Model*, *Write Excel* (or we also use *Write Database* instead of this operator). *Read Excel* reads the data that is to be used for training from a specified excel file where the data is like in Table 3 or Table 4. As it will be explained later, this data has both label and attribute information where label information is going to be predicted by classifier. *Naive Bayes* operator takes the data from *Read Excel*

and constructs a learning model according to the Naive Bayes algorithm. On the other hand, *Read Excel(2)* operator also reads a data from another excel file(or sheet) but this data only contains regular attributes since the label column is unknown for this data. *Apply Model* operator takes two inputs, which are a learned model(mod) and an unlearned data(unl), tries to categorize this unlearned data according to the learned model constructed by *Naive Bayes* operator and outputs all predictions with their probability. Actually it gives a probability score for every possible label, but it selects the label which has the highest probability as the prediction. Lastly *Write Excel* operator transfer all outputs obtained from *Apply Model* operator into another specified excel file.

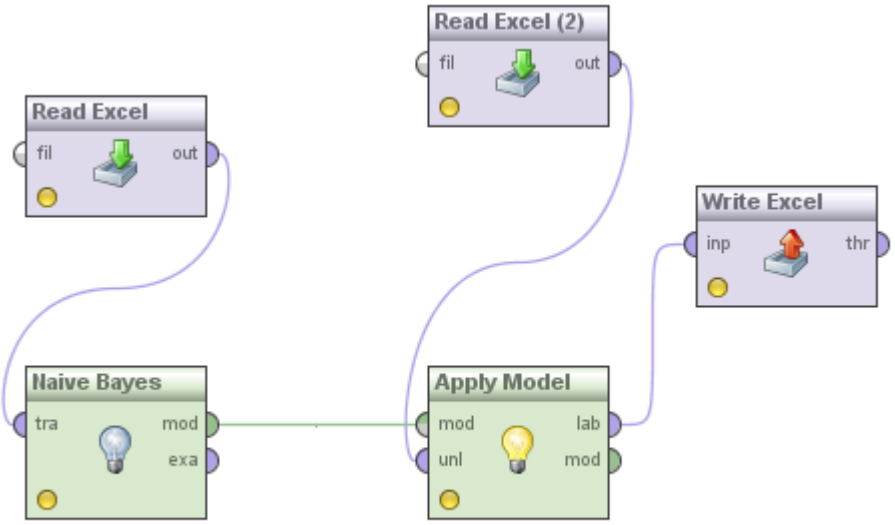

Figure 2: Overall schema of categorization process in RapidMiner

For the cross-validation part, as given in figure 3, it has only two operators which are *Read Excel* and *X-Validation* differently from the classification technique explained above. It only requires one data (one excel file) like in Table 3 since cross-validation technique select its own test and training data according to the k value as we explained in Section 2.7. We select k value as 10 due to some experimental reasons. *X-Validation* is a nested operator which means it has some sub-processes inside it as it is shown in Figure 4. For the training section, we again need to use a learning operator where we select *Naive Bayes* operator; and in the testing part, there are two operators which are *Apply Model* and *Performance*. *Apply*

Model works in the same way as mentioned before for every iterative part of Cross Validation. On the other hand, *Performance* operator calculates the *PerformanceVector* which contains some accuracy information for the testing process. A basic example of PerformanceVector output is given in Figure 5. It is possible to find the overall accuracy of the test as well as the accuracy of each class precision.

Figure 3: Overall schema of X-Validation process in RapidMiner

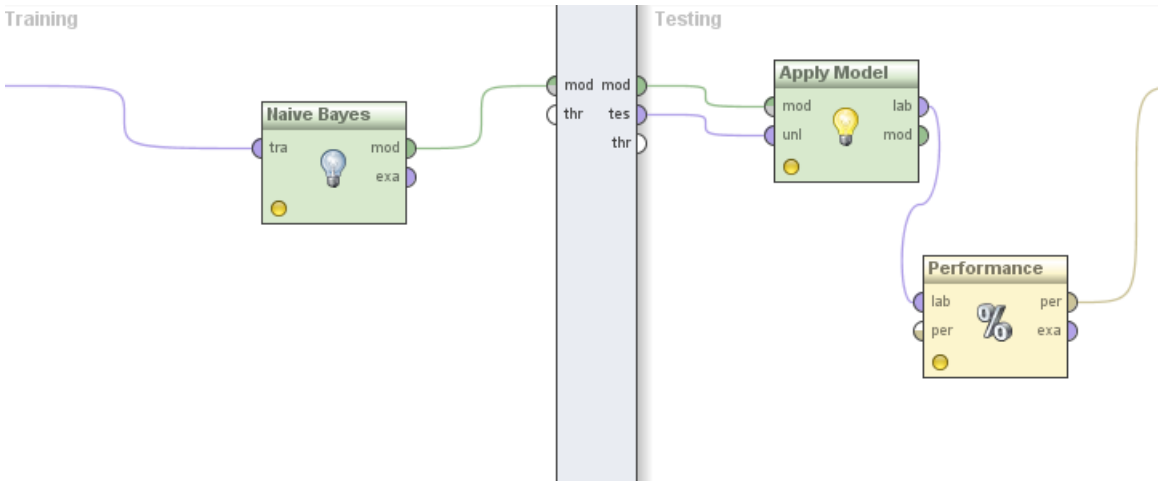

Figure 4: Sub-operators inside the X-Validation operator

This section will provide the results of the experiments along with showing the necessary steps to get better results. Initially, it is crucial to say that we have 900000 tweets to make a model but we could not train all of the data since Google spell checker has an online limitation for connections we made for checking. For that reason, we have trained a subset of our

PerformanceVector (Performance)

xt View Annotations

Multiclass Classification Performance Annotations

Table View Plot View

accuracy: 90.00% +/- 30.00% (mikro: 90.00%)

	true 90s	true morganstreet	class precision
pred. 90s	5	1	83.33%
pred. morganstreet	0	4	100.00%
class recall	100.00%	80.00%	

Figure 5: PerformanceVector

data (2000 tweets) to observe the initial results.

As the first step, we decided to take into consideration only edit distance information between correct and incorrect words; and not considering specific error types like insertion, deletion or substitution. In that case, when we have a look at the initial results, we observe that only 20 percent of the authors of unknown tweets are predicted correctly which is not good. In fact, when we think about the problem after we see the results and seek for the reasons of low accuracy, we decided to add more detailed information as we noticed that only keeping edit distance metric for spell errors in a tweet is not enough for a good learning model. For instance, if some specific user writes *tyrn* instead of *turn*, we can easily detect this word written incorrectly and we calculate Levenshtein edit distance between *tyrn* and *turn* which is 1. Just think about another user who writes *trn* instead of *turn*. In that case, again we are able to detect misspelled word and calculate edit distance which is again 1. Under these information and conditions, when we try to train the data, we make a learning model which evaluates *tyrn* and *trn* as the same type of error. However, it is obvious that they are not the same type of error and somehow we need to separate them.

It is possible to find a small part from initial data which considers only edit distance information between correct and incorrect words in table 3. In this table, we have only 4 columns which are *username*, *misspelledWord*, *correctWord*, *editDistance*. Every row in the data indicates one mistake which is typed by a user. Needless to say, a user possibly have more than

username	misspelledWord	correctWord	editDistance
90s	blessid	bless id	1
90s	madonna	Madonna	1
90s	shoop	shop	1
90s	montell	mantel	2
morganstreet	willdwliver	wildlife	4
morganstreet	kernal	kernel	1

Table 3: A small part of the first version of data

one row in the data since a user generally types lots of misspelled word when we consider the all tweets of this user. Although, it is possible to understand the meanings of columns by just looking their names, we need to explain the columns to make it clear. *username* donates user id of the user, *misspelledWord* denotes the word which is typed incorrectly by user, *correctWord* denotes the word which is the correct version of the misspelled word (we explained how we detect correct version of the word above), and *editDistance* denotes the edit distance between correct and misspelled word (we also explained how we calculate edit distance in Section 2.2). When we want to train this data, we evaluate *misspelledWord*, *correctWord*, *editDistance* columns as attributes while *username* is a label since we are trying to predict authors of tweets.

After the results and the observations of the first step, beyond calculating edit distance between words, we additionally determined error types of incorrect words and used these error types for learning model. Consider again *tyrn* and *trn* words which are typed by different user. As we have stated before, both two user intended to write *turn* and both two incorrect words has 1 as edit distance measure. However, the operation to convert *tyrn* into *turn* is a substitution of *y* and *u*. On the other hand, an deletion(*u*) operation is needed to get *trn* from *turn*. In that situation, we can distinguish users better and it constructs a better learning model for classification. We again apply this experiment on 100 users where each of users has at least 180 tweets and the results show that we have a 52 percent of accuracy for exact prediction of users.

This data which contains more detailed information for errors can be found in Table 4.

username	misspelledWord	correctWord	editDistance	insert	delete	subs1	subs2
90s	blessid	bless id	1	empty			
90s	madonna	Madonna	1			M	m
90s	shoop	shop	1		o		
90s	montell	mantel	2			a	o
90s	montell	mantel	2		l		
morganstreet	willdwliver	wildlife	4		l		
morganstreet	willdwliver	wildlife	4		w		
morganstreet	willdwliver	wildlife	4			f	v
morganstreet	willdwliver	wildlife	4		r		
morganstreet	kernal	kernel	1			e	a

Table 4: A small part of the second version of data

In this data, apart from *username*, *misspelledWord*, *correctWord*, *editDistance* columns, there are four other columns which are *insert*, *delete*, *subs1*, *subs2*. *insert* column indicates the necessary character which should be inserted to misspelled word to reach correct word. *delete* column similarly indicates the necessary character which should be deleted from misspelled word. *subs1* and *subs2* columns denotes that the character in *subs2* should be substituted with the character in *subs1* to reach again correct word. Additionally, we need to emphasize that each row in Table 4 refers to only one misspell operation in a misspelled word. However, sometimes we need more than one misspell operation to convert an incorrect word into the correct version. In such a case, there are more than one rows for this misspelled word where each row indicates a different operation. One can ask that why we did not keep all operations in one row for a specific word. The reason behind that if we use such a technique, then order of the mistakes would be important. For example, assume that a specific user u typed a word incorrectly and this word needs two substitutions which are $a-e$ and $m-n$ respectively. Then assume that we keep this information in our data as *subs1* is a, *subs2* is e, *subs1'* is m and *subs2'* is n. The problem is that the classifier would except that user u should repeat $m-n$ substitution error in *subs1'* and *subs2'* columns to identify this user. However, when user u types a word which needs $m-n$ as the only substitution or as the first substitution, then m will be located in *subs1* and n will be located in *subs2* columns. In that case, classifier may not be able to identify this user, although $m-n$ substitution error type became as a behavior of this user u.

Although this accuracy result does not seem to be evaluated as high success, it can still be considered to be a privacy leak and we can state two important points which should be emphasized. Firstly, we have a dramatic improvement when we compare it to the first step of the experiments. Success rate jumped to 52 percent from 20 percent when we integrate the specific error types into the learning model for classification. This deduction shows that error types can be defined as user behaviors that we can distinguish users by just looking their misspelling error types with specific letters, characters and words. Second important point is that results of the experiments can be improved by analyzing Naive Bayes scores. As it is known, Naive Bayes is a probabilistic classifier which returns a probability score for each class and it selects the class which has the highest probability as prediction. In the experiments, when calculating accuracy score, we look at this predictions and compare it to real class values to understand whether Naive Bayes predicted correctly. As we stated before, we could manage to predict 52 percent of users correctly. However, when we look at the incorrect part of the results, the classifier had a wrong decision for the first prediction but it mostly finds out the real value in the second or third predictions which are ranked according to the probability scores. Therefore, if we look for the real values in the first three predictions of classifier, we managed to get about 72 percent of accuracy. In other words, by using classification of error types method, we can say that a specific user most probably belongs to a group which has three user inside it. Additionally, we can extend our definition with k-anonymity at this point. As we defined k-anonymity in Section 2.9, we may say that we compose a table with 3-anonymity with respect to the set of spell errors. Here, we do not use quasi-identifiers differently from original definition, instead we use spell errors.

4.2 Experiments on TF-IDF Based Weighted Cosine Similarity

In this section, we will explain some experiments with their results that supports the claim we made in the problem definition part. Suppose we have n users $\{a_1, a_2, \ldots, a_n\}$ and we do not have any multiple account information such that one specific user might have one identifiable(public) account a_j and one anonymous account a_i. We also claim that writing style of users mostly contains enough information to identify them. In particular, we only analyze the

words used by users as writing style for the identification process. Although there are some other information like word pairs, phrases; we do not analyze them.

Since there is no available tweet data on different sets of accounts for the same users, what we did was to create two sets by using the twitter data that is publicly available as simulated in Table 5. In order to create two different sets, we took the tweets of a user written in time and divided them from median into two sets of equal size (i.e., equal number of tweets) in the first experiment. We took the first part of the tweets to simulate the public accounts (P) and the second part to simulate the anonymous accounts (A). Then, in the next set of experiments, we divided tweets of a user into three sets of equal size and again we took the first part of the tweets to simulate the public accounts (P) and the third part to simulate the anonymous accounts (A). We did not consider the second part to eliminate the topic bias otherwise we risk facing the problem of a user talking about specific topics could be easily matched which would not be fair. In these experiments our aim is to observe topic independent clues for matching users in public and private accounts.

Public		**User**		**Anonymous**
P_1	$\leftarrow$	U_1	$\rightarrow$	A_1
P_2	$\leftarrow$	U_2	$\rightarrow$	A_2
P_3	$\leftarrow$	U_3	$\rightarrow$	A_3
$\cdot$	$\leftarrow$	$\cdot$	$\rightarrow$	$\cdot$
$\cdot$	$\leftarrow$	$\cdot$	$\rightarrow$	$\cdot$
$\cdot$	$\leftarrow$	$\cdot$	$\rightarrow$	$\cdot$
P_{n-1}	$\leftarrow$	U_{n-1}	$\rightarrow$	A_{n-1}
P_n	$\leftarrow$	U_n	$\rightarrow$	A_n

Table 5: Composing public and anonymous account from real users

For each account a in $P \cup A$, we merge the tweets of a to form a single document. This way each account is represented by a document formed by merging the tweets under that account.

In these experiments, we selected some specific users, and then for each user we distribute all his/her tweets into two different documents (division techniques differentiates in experi-

ments). In other words, since we have 2 documents (each document contains one half of the user's tweets) for each user, we have 2n documents in total for n users. For these documents we created word vectors with respect to TF-IDF values and then we tried to match these documents according to *Cosine Similarity* numerical measure. Thus, our similarity function $s(a_i, a_j)$ is based on *Cosine Similarity* technique. After obtaining the results, we select the most similar document D' for each document D and we formulate success rate as number of correctly selected pairs such that D' is really the other half of D, over n.

For calculating similarity score between each user, we use RapidMiner tool again. In Figure 6, the general steps of the process is shown as the main process has five operators which are *Read Excel*, *Process Documents from Data*, *Data to Similarity*, *Similarity to Data* and *Write Database*. *Read Excel* operator reads data from an excel file where each row in the excel represents all tweets of a public or of an anonymous user. The second operator *Process Documents from File* is used for creating vectors which contain TF-IDF weights of documents (rows in excel file) by using absolute prune method. This prune method ignores some of the words in the documents according to their frequencies. *Process Documents from File* operator is a general operator which has some sub-operators inside it like *Tokenize* and *Transform Cases*. *Data to Similarity* operator calculates similarity measures based on TF-IDF vectors of the documents. Then, *Similarity to Data* operator takes the role to produce a table format like in Table 6. As it can be noticed, every row in this table represents a similarity score between two users. Lastly, *Write Database* operator transforms this table into our MYSQL database.

ID_1	ID_2	sim
ID_1	ID_3	sim
.	.	.
.	.	.
.	.	.

Table 6: Table Format after *Similarity to Data* operator

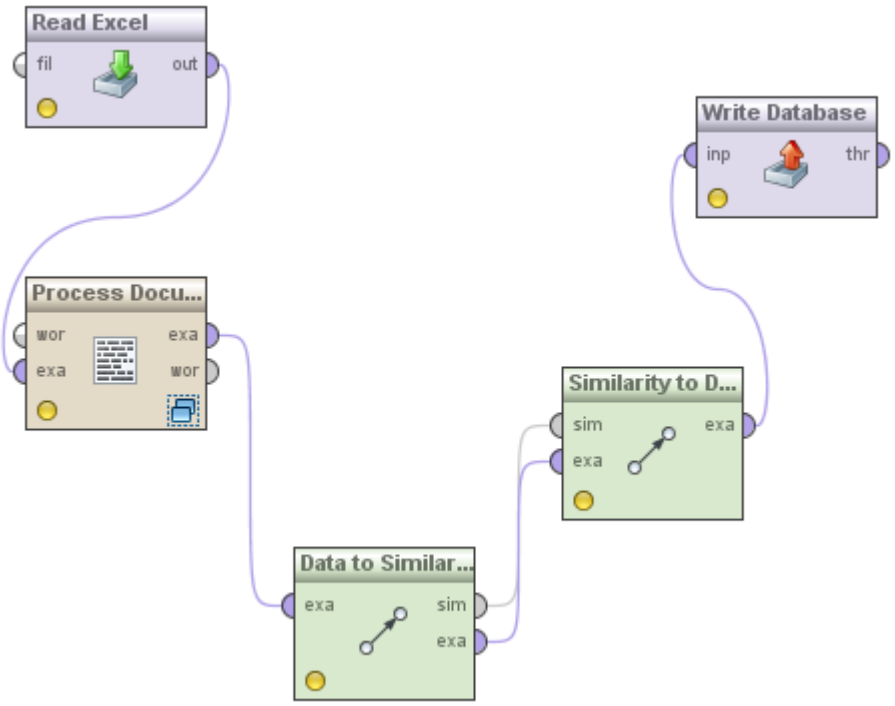

Figure 6: Cosine Similarity Process in RapidMiner

4.2.1 Experiment for users with high number of tweets and median division of tweets

In this experiments, we sorted users according to the number of tweets they have and we selected the users with high number of tweets. Additionally, to get a public and an anonymous account from a user, we divided tweets of the user from median into two equal size.

- When n is 90 or 100, each user has at least 180 tweets.
- When n is 200, each user has at least 162 tweets.
- When n is 500, each user has at least 139tweets.

where n denotes number of user in the experiment.

Assume that a specific user has 120 tweets in total. For this user, we select first 60 tweets for its public account, and last 60 tweets for anonymous account as shown in Figure 7.

For this experiment, results are given in Figure 8:

As you can notice in the results, each user has very high number of tweets, regarding this the success rates are very high too. High number of tweets means better information for a

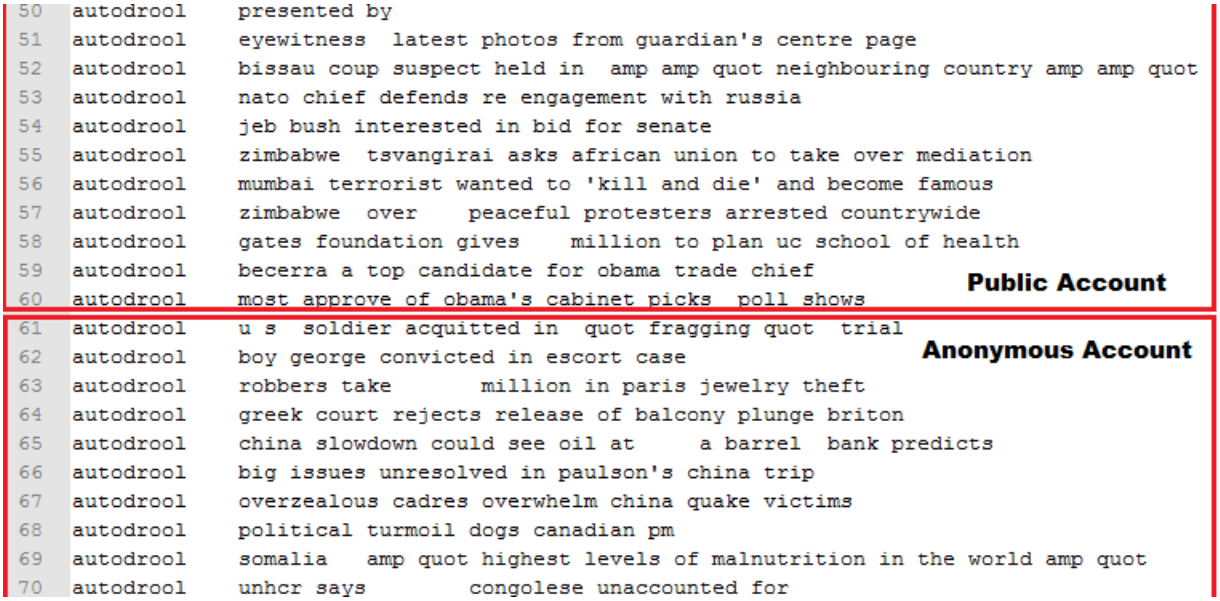

Figure 7: Division from median

user and users become more distinguishable. One other possible reason for very high match is topic dependence. In other words, when we use the median to divide the ordered data set into two halves (tweets are ordered according to time that they had been posted by user), they probably mention similar or same topics and finding a relation between two halves of a user (i.e. between public and anonymous account of a user) is more possible.

4.2.2 Experiment for users with lower number of tweets and median division of tweets

As we have pointed, one of the main reasons that the experiment in Section 4.2.1 leads very high success rates is that each user has high number of tweets which makes it possible to have a good model of the users. For this reason, we wanted to test the effect of number of tweets for the success rate when users have lower number of tweets. In this experiment, we first selected 100 users where each user has exactly 40 tweets, then we selected another 100 users where each user has exactly 50 tweets, and lastly we selected 100 users where each user has exactly 60 tweets. Note that we again use the same technique with experiment in Section 4.2.1 for creating public and anonymous accounts from a user.

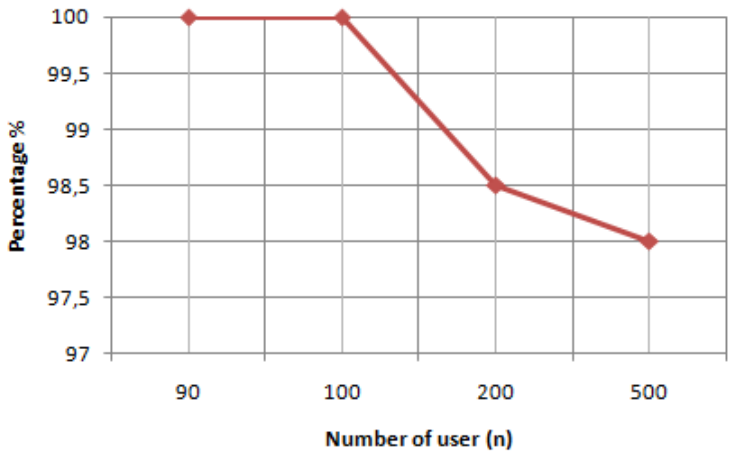

Figure 8: Percentage of correct identification of users for different number of users

When n (number of users) is always 100, the results are as in Figure 9:

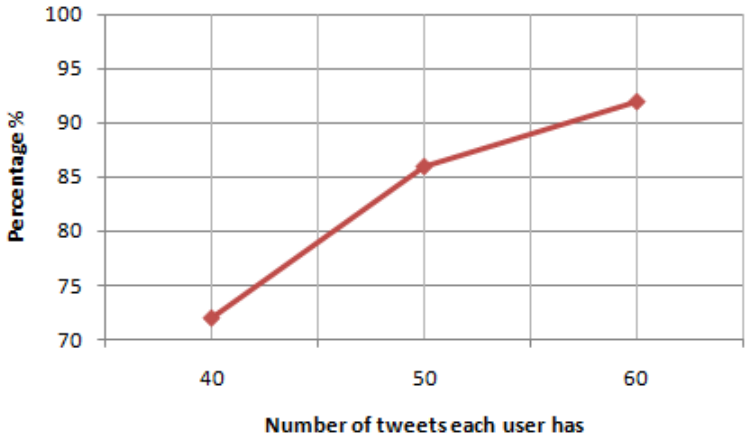

Figure 9: Percentage of correct identification of users for different number of users who have less tweets

As number of tweets per user decreases, the success rate for identification of user also decreases. The success rate is 70 percent when users have 40 tweets, however when each user has 50 and 60 tweets, the success rates jumps over 85 and 90 percent respectively. Therefore, we experimentally show that high number of tweets helps to construct a better model for

users to distinguish them between each other.

4.2.3 Experiment for different number of users without topic dependency

One other possible reason for very high match in the experiment of Section 4.2.1 is topic dependency as it is stated before. To eliminate topic dependency, initially we selected n users who has maximum number of tweets, then we selected first 1/3 part and last 1/3 part of tweets for each user where tweets are sorted according to the time that they have been posted.

Again assume a specific user who has 120 tweets in total. We select first 40 tweets together as document of public account and we select last 40 tweets for the anonymous account as shown in 10.

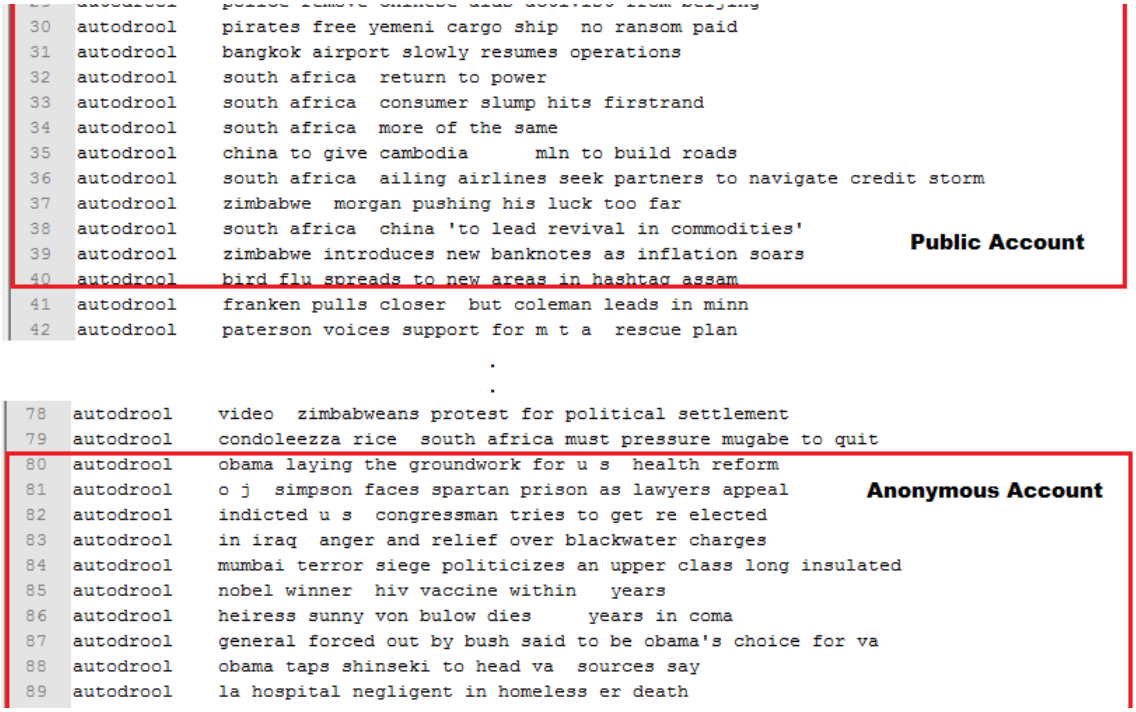

Figure 10: Division for topic independence

Consequently, we had $2n$ documents where each document contains 1/3 of tweets of a user that these tweets can be either the earliest 1/3 part of tweets of a user or the last 1/3 part of tweets of a user as shown in Figure 10. Results of this experiment are presented in Figure 11.

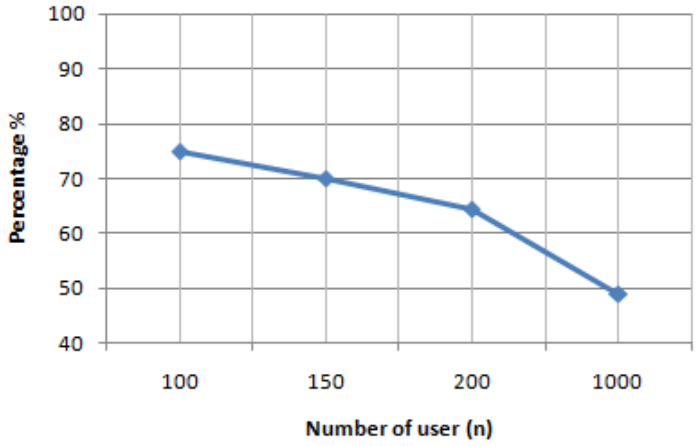

Figure 11: Exact match rate for topic independence case

We apply this process on different number of users such that user numbers are 100, 150, 200 and 1000 respectively. When we evaluate this experiment, we try to prevent topic based bias for tweet matching. Even if we choose tweets of a user from different times, it means that this user probably mentioning about different topics in his/her tweets but we are still able to match correct documents of a user with high ratios.

4.2.4 Experiment for identifying users in top k

We claim that we can match an anonymous account a_i, where $a_i \in A$, with identifiable(public) account a_j, where $a_j \in P$, of same user in top k with high accuracy where k is not too large (3,4, ...), instead of using exact match (i.e k=1). In this concept, we used a similar preprocessing technique like we used in Section 4.2.3. Again we select the least recent and the most recent tweets of each user in two different documents to prevent topic biased dependence. Afterwards, for each document d, we get an ordered list of similar documents according to scores obtained from the similarity function and we figure out the rank of the real half of the document d. After we obtain this rank for each user, we take the average of these values to determine average k. At this point, we say that we can match an anonymous account a_i with an identifiable account a_j of the same user averagely in top k.

For different number of users, the average of the k values for each test given in Figure 12:

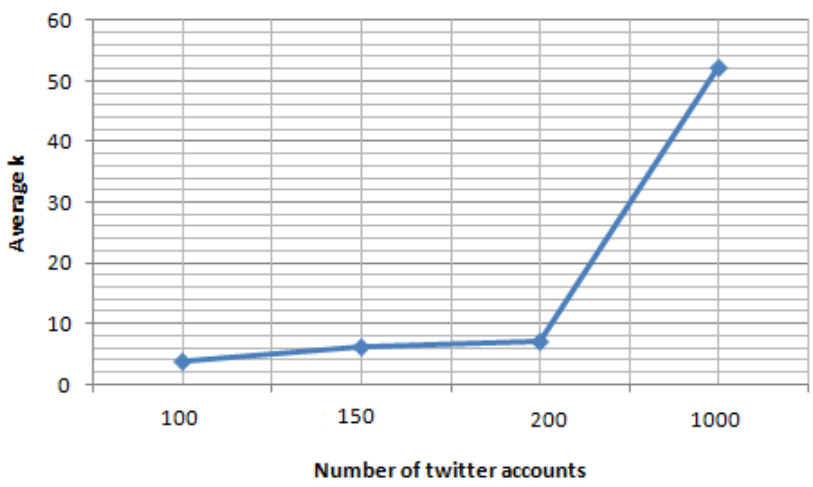

Figure 12: Average values of k for the different number of twitter accounts

Figure 12 shows that we can mostly identify anonymous accounts in top k where k is small as our claim says. We managed to identify 100, 150 and 200 users averagely in top 4, 6 and 8 respectively. However, when number of users is 1000, average value jumps to 52 because of some outliers which have big values. For that reason, we believe that it is useful to analyze distribution of k for different n values.

When $n = 100$, the distribution of k is given in Figure 13. 86 users can be identified in top k where $k = 3$.

When $n = 150$, the distribution of k is given in Figure 14. 133 users can be identified in top k where $k = 4$.

When $n = 200$, the distribution of k is given in Figure 15. 163 users can be identified in top k where $k = 3$.

When $n = 1000$, the distribution of k is given in Figure 16. 711 users can be identified in top k where $k = 8$.

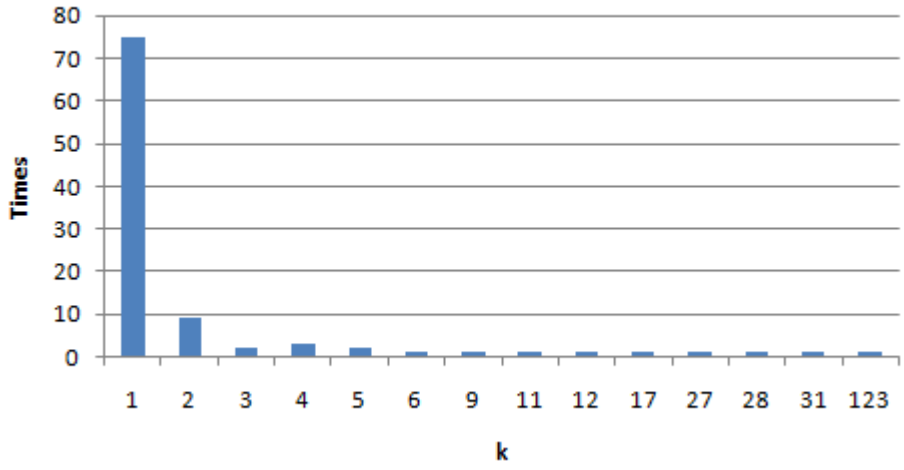

Figure 13: Distribution of k when the number of users is 100

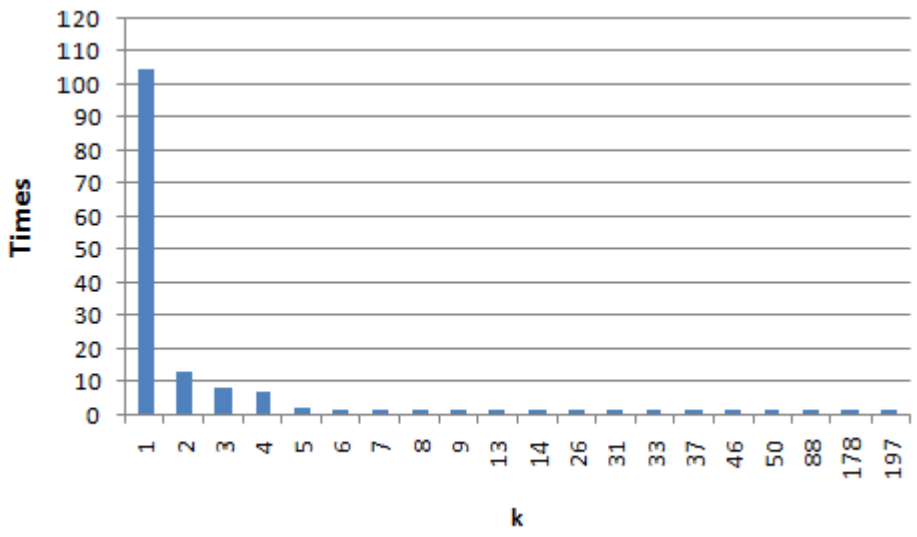

Figure 14: Distribution of k when the number of users is 150

Consequently, these figures and experiments show that users can be matched with their true identity with high accuracy in top-k where k changes depending to number of users in tests. However, experiments also show that k value is not so large which means that we can speak of a violation in privacy of individuals.

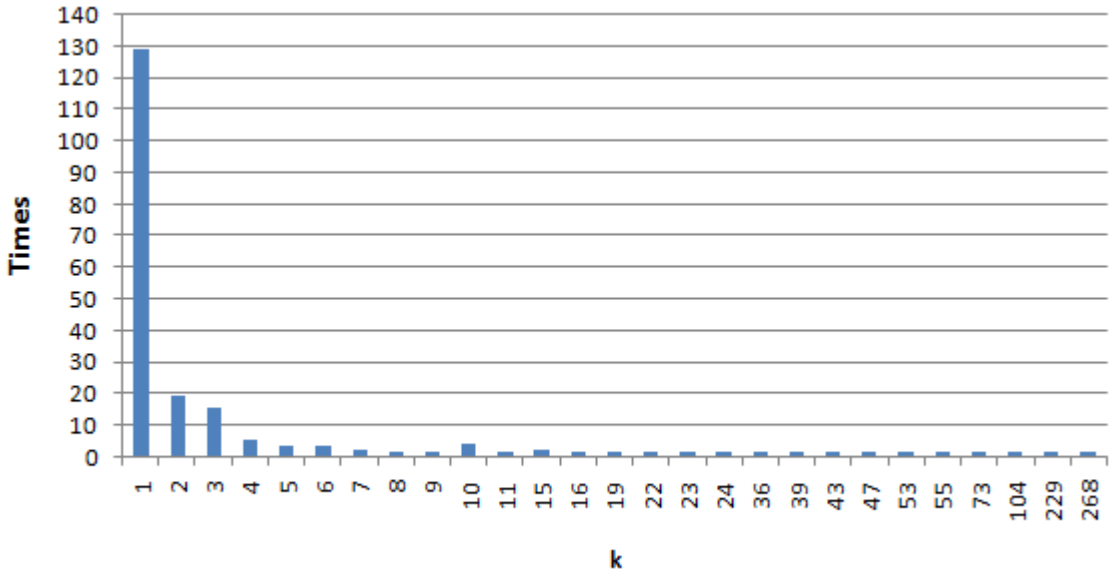

Figure 15: Distribution of *k* when the number of users is 200

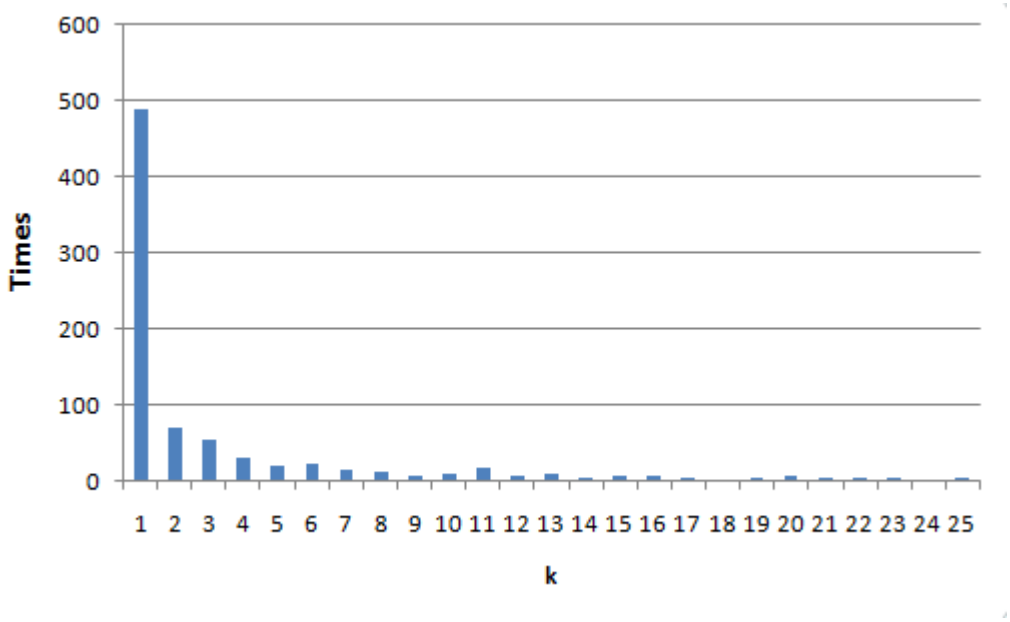

Figure 16: Distribution of *k* when the number of users is 1000

5 Related Work

In this thesis, we are working on keeping privacy of individuals by also mentioning some privacy preserving techniques like k-anonymity in Section 2.9. In recent years, privacy preserving data publishing is one of the hot research areas that researchers are interested in. One reason that points the importance of data publishing is finding a linkage through some publicly available data. Imagine that you have a hospital data which some anonymization process is applied on this data for keeping sensitive information hidden. However, some adversary can find some common quasi-identifiers from another publicly available data to identify individuals or sensitive information as this process is named as *record linkage attack*. One other way for revealing sensitive information is *Functional Dependencies(FD)* attack as Wang and Liu mention in [38]. They emphasize that existing techniques are not enough to provide privacy against FD-attacks on published data. A *functional dependency* is defined in [38] as follows:

Definition 5.0.1 *Given two attributes X and Y, a published data D satisfies the functional dependency F: $X \rightarrow Y$ if for every row(tuple) $r_1, r_2 \in D$, if X attributes of rows(or record) r_1 and r_2 are same then Y attributes of r_1 and r_2 are also same. Here, X is called as determinant attribute while Y is dependent attribute as in [38]. They consider Full Functional Dependency(FFD) in [38] as FFD is the function F: $X \rightarrow Y$ that satisfies for all X and Y values.*

FDs can be efficiently used by adversaries to violate the privacy as we said before. Consider the Table 7,8,9 which are represented in [38] to give an example of FD attack. Note that *Sex* and *Zip* attributes are quasi-identifiers(QI) while *Phone* and *Disease* are sensitive

attributes of this table. And also *Name* attribute is an ID such that it should be kept hidden for individuals' privacy. Assume that there is a functional dependency in Table 7 F: $Phone \rightarrow Zip$, where this FD means that if telephone numbers of two records are same, then the zip codes of these records are also same.

Name	**Sex**	**Zip**	**Phone**	**Disease**
Alice	F	07921	1111111	Ovarian cancer
Bob	M	07920	2222222	Bronchitis
Calvin	M	07902	3333333	Diabetes
Doris	F	07901	1000001	Ovarian cancer
Eve	F	07902	3333333	Bronchitis
Flora	F	07903	2000001	Pneumonia

Table 7: (a) The original data

When we look at Table 8, it is an anonymized table such that *Name* attribute is removed while *Sex* and *Zip* attributes generalized for satisfying 3-anonymization. First three records in this table are indistinguishable between each other with respect to quasi-identifiers(*Sex* and *Zip*) and last three records are also indistinguishable between each other.

Sex	**Zip**	**Phone**	**Disease**
*	079**	1111111	Ovarian cancer
*	079**	2222222	Bronchitis
*	079**	3333333	Diabetes
F	0790*	1000001	Ovarian cancer
F	0790*	3333333	Bronchitis
F	0790*	2000001	Pneumonia

Table 8: (b) The 3-anonymized table

However, if an adversary knows the functional dependency F: $Phone \rightarrow Zip$, then he/she can notice that the telephone numbers of the third and the fifth records are same. Therefore, their zip codes must also be same according to the FD and zip code of the third record can be changed as in Table 9. In this case, adversary managed to break 3-anonymization in Table 8 and this situation is a dangerous attack which should be a serious concern. For this problem, Wang and Liu define a privacy model which is $(d, l) - inference$, they also define some efficient algorithms to anonymization for privacy issues but they also concern about the quality

of the data that minimum loss of information should be provided while anonymization process is being applied.

Sex	Zip	Phone	Disease
*	079**	1111111	Ovarian cancer
*	079**	2222222	Bronchitis
*	0790*	3333333	Diabetes
F	0790*	1000001	Ovarian cancer
F	0790*	3333333	Bronchitis
F	0790*	2000001	Pneumonia

Table 9: (c) Table after FD inference

Machanavajjhala et al. [25] also mentions about publishing data about individuals with pointing techniques and problems about these techniques. They claim that k-anonymity is not strong enough to keep privacy of individuals such that an adversary with some background information can reveal sensitive information of individuals even if the data is protected by k-anonymization. To prove this claim, they simulate some attacks on k-anonymous dataset which are *homogeneity attack* and *background knowledge attack* by showing their practical applications for bringing a k-anonymous dataset into disrepute. For all these reasons, l-diversity privacy technique is introduced in [25, 8] as the following:

Definition 5.0.2 *A table T is l-diverse if for each equivalence class E, the probability of occurrence of all sensitive values is should be at most 1/l.*

To give a similar example like in [25, 8], consider the following Table 10. In this table *Age*, *City* and *Sex* are quasi-identifiers while *Disease* is a sensitive attribute. It is easy to notice that Table 10 is 2-anonymous with respect to quasi-identifiers. However, it is still doubtful whether privacy is preserved by k-anonymity. Consider a male who is 25 years old and an adversary wants to learn the disease of this male by assuming our dataset contains information about mentioned male. If we look for a record where *Age* is 25 and *Sex* is M, then we can reveal that this person has Leukaemia as disease; and ofcourse the adversary can reveal too.

Age	City	Sex	Disease
[20-30]	Izmir	M	Leukaemia
[20-30]	Izmir	M	Leukaemia
[30-40]	Istanbul	F	Broken Leg
[30-40]	Istanbul	F	Flu

Table 10: A table with 2-anonymity

On the other hand, when we look at the Table 11, we see that this table is both 2-anonymous and 2-diverse. Additionally, when we consider about the attack we defined above, it does not create any thread anymore. Shortly, Machanavajjhala et al. [25] shows deficiency of k-anonymization due to insufficient diversity of sensitive values in equivalence classes and they introduce l-diversity which provides more efficient privacy in publishing datasets.

Age	City	Sex	Disease
[20-30]	Izmir	M	Cold
[20-30]	Izmir	M	Leukaemia
[30-40]	Istanbul	F	Broken Leg
[30-40]	Istanbul	F	Flu

Table 11: A table with 2-diversity

As the part of publishing publicly available datasets, *data suppression* and *data swapping(changing)* are also some basic concepts that are mentioned in [11, 13, 14, 10, 8]. In these papers, while the privacy preserving of data is the main goal of their work, but quality of the data is also one of the indispensable conditions in the research area. For every generalization process in k-anonymity or l-diversity to provide privacy of data is loss of information in the data. In other words, there is a trade of between privacy and quality of data. Consider a data which contains thousands of records and assume that k-anonymity or l-diversity is not satisfied just because of a few number of records. In that case, we may prefer to change the contents of these records in a way that privacy preserving techniques will be satisfied. We call this operation as *data swapping*. Or we may prefer to ignore these records by deleting them and this process is called as *data suppression*.

Zhou and Pei [43] mention a different kind of attack on social work named as *Neighborhood Attack* for identification of users in a graph. They remind that structure of a social web can be represented by graphs where nodes denote users and edges denote the friendship between these users. Their main goal to show that an adversary, who has some knowledge about individuals, can attack to the graph to identify some of the victims. Similarly, Das et al. [12] considers a social network as a weighted graph and they apply *edge weight anonymization* for the privacy preservation of individuals. Backstorm et al. [4] defines a technique for identification of vertices in anonymous graphs by using isomorphism and automorphism features of graphs.

6 Conclusion and Future Work

In this thesis, we mainly mentioned that some people tend to create anonymous accounts on Twitter (or another platform on social work). The reason behind this is that they want to show a reaction towards political and social issues as social web one of the best places where users are able to reach the most number of people. Additionally, they do not want to come to light since they do not want to get in trouble. In that sense, we claimed that just creating another account with some other user name or changing IP etc. are not enough to preserve their privacy. To prove that claim, we used a real twitter data which is publicly available, and also we created public and anonymous accounts of same users to check whether we can identify an anonymous account with its true public account.

For this purpose, we introduced two different techniques to show that anonymous accounts are able to be identified. The first technique is that we created a learning model of each user through their misspell error information. We kept the information of the words which users type incorrectly, their correct versions, the edit distance between these correct and incorrect words, type of the misspells and the specific characters for each mistake. In the long term, we observe that these errors became as a behavior of users and we can match an anonymous account with its public account through this learning model by using Naive Bayes classification algorithm.

As the second technique, for each user, we collected all tweets sequentially of that user as one document and then for each user again, we created vector of weights where every index is tf-idf value of a term. When we want to match a specific anonymous account with a public

account, we used cosine similarity technique which returns a similarity score for each public account. Then, we rank these public account according to the similarity values and we select the most similar k public account to match.

Experiments show that both two techniques had high accuracy for matching an anonymous account with a public account in top k where k is small. Even for the exact matching (for $k = 1$), we succeeded to reach good accuracy and we obviously had much better results for the different values of k (values for bigger than 1).

For future work part, we suggested two different solutions in Section 3.3 which should be applied and tested. For the misspell identification technique, we suggested to refine misspell words with their correct versions. On the other hand, for the cosine similarity document matching technique, we advised to replace the words, of which tf-idf values are high, with their synonyms. A tool may be implemented to apply these solutions and the results should be tested with some number of experiments.

One other thing which can be tested in the following works is that considering the specific time intervals of tweet for matching process. In other words, when an anonymous user is desired to be identified, then some tweets of anonymous and public users in a specific time interval may be used. In that case, because of the probability that anonymous and public users may talk about similar topics, some interesting results can be obtained to compare with the current results.

References

[1] C. Apté, F. Damerau, and S.M. Weiss. Automated learning of decision rules for text categorization. *ACM Transactions on Information Systems (TOIS)*, 12(3):233–251, 1994.

[2] M. Arrington. AOL Proudly Releases Massive Amounts of Private Data. http://techcrunch.com/2006/08/06/aol-proudly-releases-massive-amounts-of-user-search-data/, 2006. [Online; accessed 22-July-2012].

[3] M. Baake, U. Grimm, and R. Giegerich. Surprises in approximating levenshtein distances. *Arxiv preprint q-bio/0601006*, 2006.

[4] L. Backstrom, C. Dwork, and J. Kleinberg. Wherefore art thou r3579x?: anonymized social networks, hidden patterns, and structural steganography. In *Proceedings of the 16th international conference on World Wide Web*, pages 181–190. ACM, 2007.

[5] M. Barbaro and T. Zeller. A Face Is Exposed for AOL Searcher No. 4417749. http://www.nytimes.com/2006/08/09/technology/09aol.html?_r=1, 2006. [Online; accessed 22-July-2012].

[6] D. M. Blei. Naive bayes. COS424 Lecture Notes, Princeton University, 2008.

[7] H. Chen. String metrics and word similarity applied to information retrieval.

[8] E.A. Çiçek. Ensuring location diversity in privacy preserving spatio-temporal data mining. 2009.

[9] V. Ciriani, S. Capitani di Vimercati, S. Foresti, and P. Samarati. κ-anonymity. *Secure Data Management in Decentralized Systems*, pages 323–353, 2007.

[10] L.H. Cox. Suppression methodology and statistical disclosure control. *Journal of the American Statistical Association*, pages 377–385, 1980.

[11] T. Dalenius and S.P. Reiss. Data-swapping: A technique for disclosure control. *Journal of statistical planning and inference*, 6(1):73–85, 1982.

[12] S. Das, Ö. Egecioglu, and A. El Abbadi. Anonymizing edge-weighted social network graphs. *Computer Science, UC Santa Barbara, Tech. Rep. CS-2009-03*, 2009.

[13] P. Diaconis and B. Sturmfels. Algebraic algorithms for sampling from conditional distributions. *The Annals of statistics*, 26(1):363–397, 1998.

[14] G.T. Duncan and S.E. Fienberg. Obtaining information while preserving privacy: A markov perturbation method for tabular data. In *Joint Statistical Meetings*, pages 351–362, 1997.

[15] N. Fuhr, S. Hartmann, G. Knorz, G. Lustig, M. Schwantner, and K. Tzeras. *AIR/X-a Rule Based Multistage Indexing System for Large Subject Fields*. Citeseer, 1991.

[16] E. Hargittai. Hurdles to information seeking: Spelling and typographical mistakes during users online behavior. *Journal of the Association for Information Systems*, 7(1):52–67, 2006.

[17] A. Jesdanun. AOL: Breach of Privacy Was a Mistake. `http://www.washingtonpost.com/wp-dyn/content/article/2006/08/07/AR2006080700790.html`, 2006. [Online; accessed 22-July-2012].

[18] R. Jones, R. Kumar, B. Pang, and A. Tomkins. I know what you did last summer: query logs and user privacy. In *Proceedings of the sixteenth ACM conference on Conference on information and knowledge management*, pages 909–914. ACM, 2007.

[19] D. Koller and M. Sahami. Hierarchically classifying documents using very few words. 1997.

[20] K. Kukich. Techniques for automatically correcting words in text. *ACM Computing Surveys (CSUR)*, 24(4):377–439, 1992.

[21] D. Lewis. Naive (bayes) at forty: The independence assumption in information retrieval. *Machine Learning: ECML-98*, pages 4–15, 1998.

[22] D.D. Lewis and M. Ringuette. A comparison of two learning algorithms for text categorization. In *Third annual symposium on document analysis and information retrieval*, volume 33, pages 81–93, 1994.

[23] D.D. Lewis, R.E. Schapire, J.P. Callan, and R. Papka. Training algorithms for linear text classifiers. In *Proceedings of the 19th annual international ACM SIGIR conference on Research and development in information retrieval*, pages 298–306. ACM, 1996.

[24] Y.H. Li and A.K. Jain. Classification of text documents. *The Computer Journal*, 41(8):537–546, 1998.

[25] A. Machanavajjhala, D. Kifer, J. Gehrke, and M. Venkitasubramaniam. l-diversity: Privacy beyond k-anonymity. *ACM Transactions on Knowledge Discovery from Data (TKDD)*, 1(1):3, 2007.

[26] D. Mladenić. Feature subset selection in text-learning. *Machine Learning: ECML-98*, pages 95–100, 1998.

[27] I. Moulinier and J. Ganascia. Applying an existing machine learning algorithm to text categorization. *Connectionist, Statistical and Symbolic Approaches to Learning for Natural Language Processing*, pages 343–354, 1996.

[28] I. Moulinier, G. Raskinis, and J.G. Ganascia. Text categorization: a symbolic approach. In *Fifth Annual Symposium on Document Analysis and Information Retrieval*, pages 87–99, 1996.

[29] J.R. Rao and P. Rohatgi. Can pseudonymity really guarantee privacy. In *Proceedings of the Ninth USENIX Security Symposium*, pages 85–96, 2000.

[30] G. Salton and C. Buckley. Term-weighting approaches in automatic text retrieval. *Information processing & management*, 24(5):513–523, 1988.

[31] G. Salton, A. Wong, and C.S. Yang. A vector space model for automatic indexing. *Communications of the ACM*, 18(11):613–620, 1975.

[32] R.E. Schapire and Y. Singer. Boostexter: A boosting-based system for text categorization. *Machine learning*, 39(2):135–168, 2000.

[33] H. Schütze, D.A. Hull, and J.O. Pedersen. A comparison of classifiers and document representations for the routing problem. In *Proceedings of the 18th annual international ACM SIGIR conference on Research and development in information retrieval*, pages 229–237. ACM, 1995.

[34] F. Sebastiani. Machine learning in automated text categorization. *ACM computing surveys (CSUR)*, 34(1):1–47, 2002.

[35] F. Sebastiani, A. Sperduti, and N. Valdambrini. An improved boosting algorithm and its application to text categorization. In *Proceedings of the ninth international conference on Information and knowledge management*, pages 78–85. ACM, 2000.

[36] P.N. Tan, M. Steinbach, V. Kumar, et al. *Introduction to data mining*. Pearson Addison Wesley Boston, 2006.

[37] K. Tzeras and S. Hartmann. Automatic indexing based on bayesian inference networks. In *Proceedings of the 16th annual international ACM SIGIR conference on Research and development in information retrieval*, pages 22–35. ACM, 1993.

[38] H. Wang and R. Liu. Privacy-preserving publishing microdata with full functional dependencies. *Data & Knowledge Engineering*, 70(3):249–268, 2011.

[39] Y. Yang and C.G. Chute. An example-based mapping method for text categorization and retrieval. *ACM Transactions on Information Systems (TOIS)*, 12(3):252–277, 1994.

[40] Y. Yang and J.O. Pedersen. A comparative study on feature selection in text categorization. In *MACHINE LEARNING-INTERNATIONAL WORKSHOP THEN CONFERENCE-*, pages 412–420. MORGAN KAUFMANN PUBLISHERS, INC., 1997.

[41] EM Zamora, J.J. Pollock, and A. Zamora. The use of trigram analysis for spelling error detection. *Information Processing & Management*, 17(6):305–316, 1981.

[42] H. Zhang. The optimality of naive bayes. *A A*, 1(2):3, 2004.

[43] B. Zhou and J. Pei. Preserving privacy in social networks against neighborhood attacks. In *Data Engineering, 2008. ICDE 2008. IEEE 24th International Conference on*, pages 506–515. IEEE, 2008.